Accepted Manuscript

An agility-oriented and fuzziness-embedded semantic model for collaborative cloud service search, retrieval and recommendation

Daren Fang, Xiaodong Liu, Imed Romdhani, Pooyan Jamshidi, Claus Pahl

PII: S0167-739X(15)00305-2
DOI: http://dx.doi.org/10.1016/j.future.2015.09.025
Reference: FUTURE 2855

To appear in: *Future Generation Computer Systems*

Received date: 16 June 2014
Revised date: 13 July 2015
Accepted date: 19 September 2015

Please cite this article as: D. Fang, X. Liu, I. Romdhani, P. Jamshidi, C. Pahl, An agility-oriented and fuzziness-embedded semantic model for collaborative cloud service search, retrieval and recommendation, *Future Generation Computer Systems* (2015), http://dx.doi.org/10.1016/j.future.2015.09.025

Future Generation Computer Systems
journal homepage: www.elsevier.com

An agility-oriented and fuzziness-embedded semantic model for collaborative cloud service search, retrieval and recommendation

Daren Fang [a*], Xiaodong Liu [a], Imed Romdhani [a], Pooyan Jamshidi [b], Claus Pahl [b]

[a] School of Computing, Edinburgh Napier University, Edinburgh, EH10 5DT, UK
[b] School of Computing, Dublin City University, Dublin, Ireland

ARTICLE INFO

Article history:
Received
Received in revised form
Accepted
Available online

Keywords:
Cloud computing
Service agility
Semantic model
Service discovery
Ontology evolution
Knowledge retrieval

ABSTRACT

Cloud computing enables a revolutionary paradigm of consuming ICT services. However, due to the inadequately described service information, users often feel confused while trying to find the optimal services. Although some approaches are proposed to deal with cloud service retrieval and recommendation issues, they would only work for certain restricted scenarios in dealing with basic service specifications. Indeed, the missing extent is that most of the cloud services are "agile" whilst there are many vague service terms and descriptions. This paper proposes an agility-oriented and fuzziness-embedded cloud service ontology model, which adopts agility-centric design along with OWL2 (Web Ontology Language) fuzzy extensions. The captured cloud service specifications are maintained in an open and collaborative manner, as the fuzziness in the model accepts rating updates from users on the fly. The model enables comprehensive service specification by capturing cloud concept details and their interactions, even across multiple service categories and abstraction levels. Utilizing the model as a knowledge base, a service recommendation system prototype is developed. Case studies demonstrate that the approach can outperform existing practices by achieving effective service search, retrieval and recommendation outcomes.

1. Introduction

Cloud computing revolutionizes the world's ICT with on-demand provisioning, pay-per-use self-service, ubiquitous network access and location-independent resource pooling. Its reliable, scalable and elastic computational services and resource provision can adapt rapidly and effectively to nearly all kinds of needs for all major industry sectors [1, 2]. As considerable efforts are made to drive and enhance the interoperability and composition of cloud services/resources [3, 4, 5], significant research gaps are found among the proposed service reference frameworks and models. On the other hand, along with the rapid development in the field, the number of cloud services continues growing whilst the market becomes increasingly complex. Cloud service consumers (CSCs) thus, may need to dig deeply to find the optimal services, by researching on a large number of service descriptions, characteristics, properties, service level agreements (SLAs), etc. Furthermore, regarding the services' features, functionalities, customizability and interoperability, etc., existing cloud service providers (CSPs) offer a diversity of interfaces, standards, policies and SLA parameters, which result into numerous difficulties in service information retrieval, interpretation and analysis [6, 7]. Consequently, these impose urgent needs and great challenges on the specification and retrieval of cloud services, whereas an effective cloud service recommendation system is in demand for a variety of CSCs.

Recently, as a series of cloud computing/service semantic models propagate [4, 8, 9, 10, 11, 12], they suffer from certain limitations. Firstly, the majority of the existing models cannot maintain comprehensive service information across multiple abstraction levels (i.e. Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS)). These models fail to reveal the various agile interactions among cloud services and resources of such matrix structure (e.g. SaaS services can be deployed on PaaS platforms whilst PaaS services may rely on IaaS resources). Secondly, a limited number of models can effectively present the diverse full and potential service functions and features; none of them clarifies the range of connections or cooperation among cloud services and companies who have (hidden) relationships (e.g. some cloud services can orchestrate with others whilst some CSPs have certain industry relationships). Thirdly, most of the cloud services are "agile", i.e. adaptable at run time in their functions, interfaces, capacity, etc. Yet, such agility aspects are often ignored or poorly disclosed in existing models. Consequently, the lack of these critical aspects would cause ineffectiveness while implementing service search, discovery, retrieval, and recommendation tasks.

To eliminate the above limitations, a novel semantic model is proposed, notably the agility-oriented and fuzziness-embedded cloud service ontology (AoFeCSO). It adopts an agility-centric design and maximally utilizes the full range of OWL2 specifications. Moreover, AoFeCSO is deployed as a fuzziness-embedded model that stays active. It comprises fuzzy weighted service specifications to present inexplicit/controversial facts. The fuzzy weights can be collected from CSCs, CSPs and cloud service brokers (CSBs), through the form of "collaborative

service specification fuzzy ratings", i.e. users can collaborate to rate the service specification applicability. Compared with the conventional ontology building and managing processes executed by limited number of field experts, the novel collaborative ratings make AoFeCSO more resourceful as well as more credible. This would bring much more cloud computing knowledge than any single group of experts does alone.

Using AoFeCSO as a central interacting knowledge base, a collaborative cloud service search, retrieval and recommendation system (CSR) prototype is developed. With its built-in fuzzy rating management and ontology evolution mechanism, CSR facilitates automatic and dynamic model evolution without interrupting concurrent service retrieval actions. The paper's contributions are: 1) an agility-oriented and fuzziness-embedded cloud service semantic model that maintains comprehensive and in-depth service information; it ultimately comprises a diversity of cloud service descriptions, service resource aspects, characteristics and features, plus their interactions, as a single retrievable knowledge source; 2) a cloud service recommendation system that is deployed on top of the model, allowing system users to not only search and retrieve cloud services flexibly and effectively, but also participate in model contents updates, which ultimately drive dynamic model evolution.

The rest of the paper is organized as follows: Section 2 introduces the background concepts and knowledge. Section 3 defines the architecture design of the proposed semantic model. Section 4 describes the adopted fuzzy OWL2 extension technique and ontology fuzzy assertion management. Section 5 illustrates the prototype implementation and component interactions. Section 6 uses a case study to demonstrate how the proposed model captures cloud service specifications and how the relevant prototype features for cloud service search, recommendation and retrieval are provided. Section 7 evaluates the model using state-of-the-art ontology evaluation approaches. Section 8 discusses the related research regarding web/cloud service semantic model, service recommendation system and ontology fuzzy extension. Finally, Section 9 concludes the paper with summaries and future work.

2. Background

In recent years, Web Ontology Language (OWL) [13] has been widely adopted for web service semantic specifications [14, 15]. The formal entity specification and reference framework can enable the integration of a wide range of aspects, e.g. context information [16], user requirements [17], business processes [18]. Accordingly, this would assist service design, development, invocation and composition tasks in pervasive environments [6].

In fact, unlike web services and many other domains, cloud computing involves many vague and imprecise descriptions, terms, categorizations, etc. This may incur several issues during specification process. For instance, according to the majority of literature, "availability" and "security" are two separate service properties, yet some [19, 20] argue that availability is a sub category of security. For those diverse service models and characteristics, should Amazon S3, Dropbox and Google Drive be regarded as SaaS, PaaS, IaaS or Storage-as-a-Service? Do they have the same extent (degree to the capability) towards scalability, reliability, interpretability? Indeed, conventional OWL/OWL2 modeling techniques cannot handle the above scenarios effectively, since they are designed to clarify explicit knowledge with concrete axioms, either true or false [6]. Fundamentally, this is due to the formal description logical (DL) consistency requirement which does not support such fuzziness [21, 22].

Fuzzy logic (FL) is a well-known extension to DL that has been used widely in many fields. It includes two theories, known as fuzzy set [23] and fuzzy relationship [24]. The former describes vague subsumption between classes and their members, whereas the latter specifies uncertain relationships between individuals and classes. On the other hand, probabilistic logic network [25] (PLN) is another theory for uncertainty representation and inferences. It extends the existing fuzzy theories and their reasoning applicability to a great extent: the FL's fuzzy membership theory is further divided into a number of detailed scenarios (e.g. degreed belonging, chanced belonging, sharing partial properties and overall weighted judgment). The FL's fuzzy relationship theory is extended with higher-order and N-ary logical relationships.

While traditional OWL modelling techniques cannot handle and express uncertainties, FL and PLN theories is able to provide extended logic (reasoning) support for fuzzy specifications. AoFeCSO adopts OWL2 fuzzy extensions on the ground of these theories. This significantly enhances the accuracy of the model specification and expression with the most appropriate facts. More specifically, as an ordinary ontology axiom can only clarify a definite fact, a fuzzy-extended axiom can describe the fact along with "a truth degree". The degree of truth, usually a float of interval (0, 1), is viewed as the fuzzy weight of the axiom. With such weighted assertions, AoFeCSO is able to clarify a variety of vague specifications. For instance, a service owns certain "partial" properties. A service works "closely" with another service. A service is sometimes but not always regarded as what it is being specified.

3. AoFeCSO model architecture design

This section presents the design of the proposed cloud service ontology. Firstly, it introduces the loosely-coupled and agility-centric ontology design features. Subsequently, the design of the object property, data property and annotation property constructions are revealed respectively. Finally, it discusses the ontology design patterns adoption and application.

3.1. Loosely-coupled foundation

AoFeCSO is deployed with a "loosely-coupled" ontology foundation. Firstly, it adopts flexible membership classifications, which enables loose (class) boundary restrictions. Secondly, it maximally utilizes property specifications for enhanced reasoning application. More specifically, they are represented as follows:

1) In AoFeCSO, cloud services are asserted as individuals that belong to the respected cloud company classes (instead of certain cloud service models). Among those who are related, there are appropriate relationships such as "rely resource of", "have control over" and "can orchestrate with".

2) The cloud service delivery/deployment models and role/party specification are revealed via object relationships. Object property specifications are asserted from a cloud service towards its respected service model/role classes, e.g. service A "is delivered as" IaaS; service B "is deployed as" public cloud; company C "is recognized as" CSP. In this way, in AoFeCSO, a service may own multiple models and roles, e.g. both IaaS and SaaS, both public and private cloud, or even both CSC and CSP.

3) The characteristics and properties that cloud services apply are illustrated as they have some relationships with the sub entities of main service attribute classes, e.g. service characteristics (elasticity, adaptability, reliability, etc.) and service features (monitoring, notification, multiple OS and programming language support, migration and transition support, etc.).

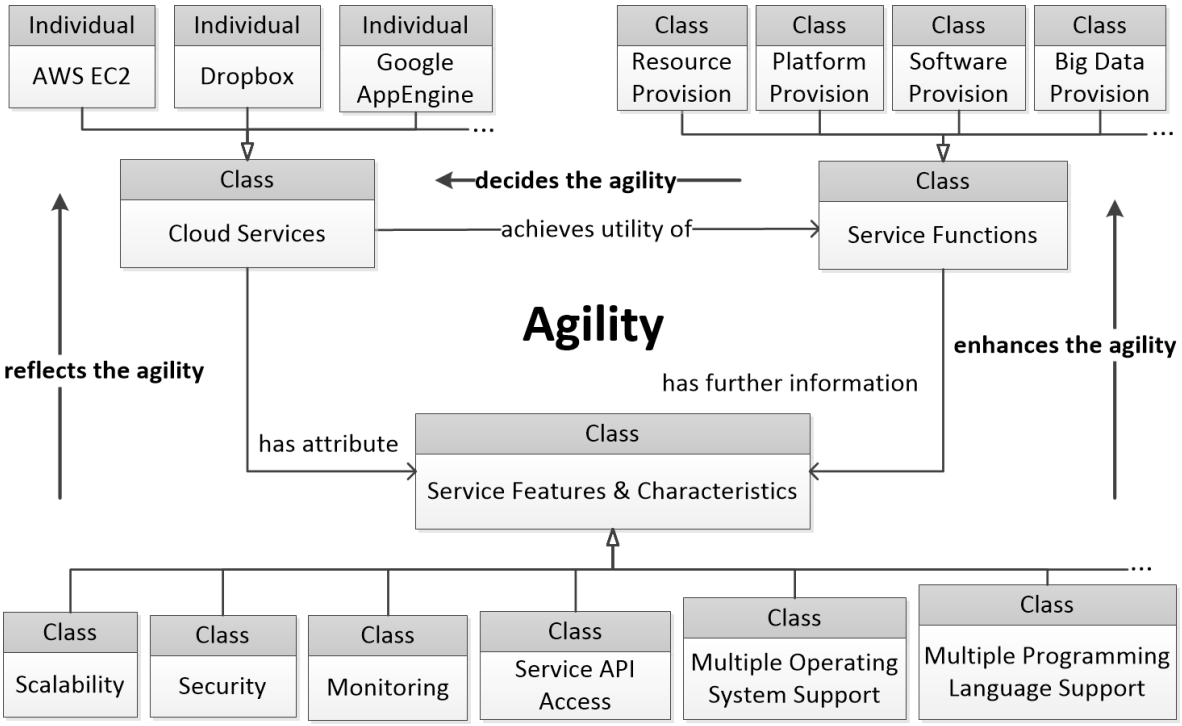

Fig. 1. Agility-centric Ontology Design

4) In AoFeCSO, a cloud services can own several functions as well. Except of the main designed function(s), a service is often specified with additional functions as long as it can serve the purpose. For instance, IaaS compute services may also provide application development platform, or even network, database and storage functions.

3.2. Agility-centric design

In cloud computing, agility is generally referred as the ability of a cloud service to react appropriately and rapidly to certain (adaptation, customizability, interoperability, etc.) requirements [26]. In fact, such reaction capability may counts on a diversity of service elements, such as the service deployment, flexible resource provision, comprehensive monitoring, notification, orchestration supports, etc.

Fundamentally, the functions a service achieves should matter the most regarding one's agility, since different functions require distinct architecture designs and resource provisions [1, 26]. Most of the SaaS services, for instance, rely on fairly limited computational resources and provide single or very limited functions. Meanwhile, typical PaaS services do not necessarily have fixed application-scale functions; instead, they are often to develop or deploy certain applications/services where certain (potential) usage/functions can be achieved. Similarly, for those IaaS services which are designed for general computing needs, they would offer greater service control, access and customization whilst they can achieve even more (potential) usage purposes. Indeed, the ranges of functions and resources a cloud service is deployed decide its agility during service composition. Accordingly, agility inevitably becomes the link while specifying the above service function aspects and their potential interactions.

The various cloud service characteristics and features can be viewed as the further information regarding one's main and potential service functions [1, 26]. Elasticity and scalability, for instance, are typical cloud service characteristics. Their sub-concepts (e.g. available VM sizes, scaling options, further details of vCPU speed/cores, intranet/Internet connection speed, memory and virtual storage sizes, etc.) are, in fact, detailing a service's capability of scaling, either up/down or in/out as required. Therefore, elasticity and scalability are extremely relevant to cloud service agility. Likewise, the ability to support different platforms, OSs, programming languages and application programming interfaces (APIs) can reflect cloud service agility. The supporting platforms, OSs, programming languages and APIs clearly state a service's interoperability and configurability towards its agility. Similarly, detailed notification, monitoring and security features can be considered relevant to cloud service agility. Notification basically comprises the different service usage notifications and various service health notifications. Monitoring consists of a diversity of service element notification, log monitoring, performance monitoring, and security monitoring. Cloud security aspects are generally divided into access control and data security. Access control comprises the different layers that a cloud service supports for its security implementation, e.g. application layer, data layer, network layer, process layer and system layer [27, 28]. Data security involves the data encryption and management supports for its security implementation, e.g. client/application encryption, data loss prevention, database encryption, externally managed encryption, file/folder encryption and digital rights management, instance managed encryption, link network encryption, and provider managed encryption, proxy encryption [29, 30]. Indeed, all these aspects above are often deployed as the guarantee for cloud service agility requirements, since they ensure the availability, reliability, integrity, confidentiality for appropriate agility responses. Consequently, a wide range of service characteristics and features are seen as the detailed reflection of cloud service agility.

As a result, illustrates in Fig. 1, agility becomes the bridging aspect that incorporates cloud service functions, characteristics and features, both functionally and non-functionally. To this extent, agility becomes the overall reflection of a cloud service's profile and capability. This is how AoFeCSO models cloud service specification by focusing the in-depth cloud service concept details and their relationships.

3.3. Ontology construction

Built on the ground of the existing cloud computing/service models and knowledge, AoFeCSO adopts the full range of OWL2 property assertions, where several property handling techniques are employed. Fig. 2, Fig. 3 and Fig. 4 demonstrate the extensions AoFeCSO achieved in contrast to other existing models (i.e. [8, 9, 10, 11, 12, 31]).

3.3.1 In-depth cloud service object property assertion

In ontology, an object property declares a certain relationship between two entities. While existing practices [4, 9] utilize such for attributing cloud service characteristics, functional and non-functional properties, very few touches the details of how or how well those cloud services own these characteristics and properties.

Shown in Fig. 2, AoFeCSO describes the lower-level details regarding the service characteristics and features. For instance, scalability is divided into vertical scalability and horizontal scalability, where each of them has individual sets of concepts. Security comprises access control and data security; each category leads to own sets of security aspects [19, 29]. By digging into the details and relating them with appropriate cloud services, AoFeCSO is capable of expressing in-depth facts of cloud services' characteristics, features and functions.

3.3.2 Explicit cloud service and concept relationship assertion

Existing models [4, 8, 9, 10, 11, 12, 32] tend to ignore the many relationships among cloud computing concept. Firstly, the interoperability between CSPs and cloud services is often inexplicitly expressed. Indeed, many CSPs are found with certain industry connections; several cloud services are built with the ability to interact agilely with others. Secondly, interactions among certain cloud service properties are often missed. In fact, there are several obvious/hidden relations among cloud service characteristics, features and functions. For instance, scalability is often attributed to elasticity to a certain extent; monitor features may affect services' scaling and load balancing behaviors.

Demonstrated in Fig. 2, AoFeCSO covers these relationships via individual-to-individual, class-to-class and individual-to-class object property assertions. Among such relevant cloud services, companies and other concepts, various direct/indirect and strong/weak relationships are explicitly revealed (e.g. "has industry relationship with", "is controlled by", "affects", etc.) As these object properties are asserted with property characteristics such as "transitive", "symmetric" and "inverse property", it

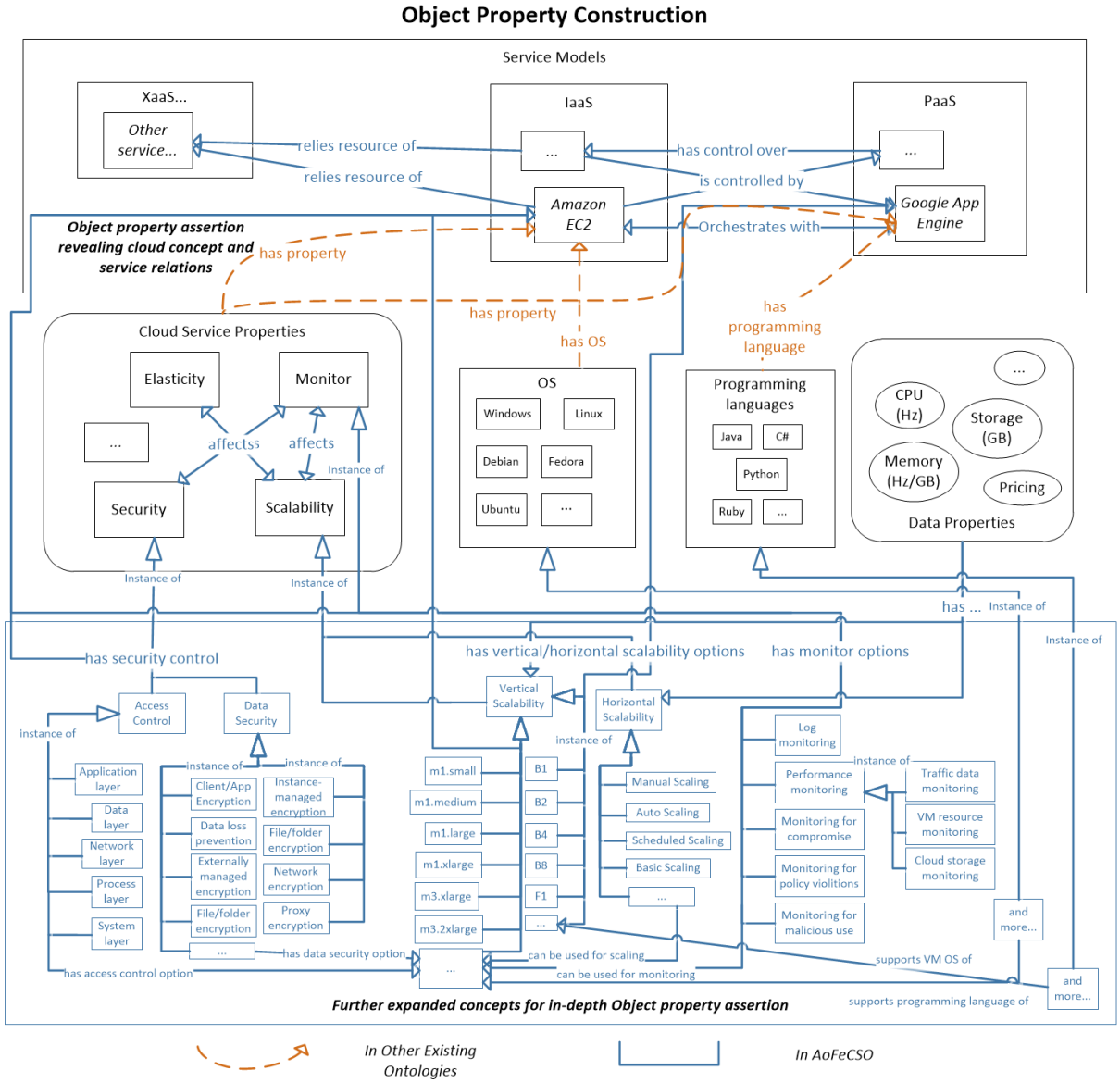

Fig. 2. Advances of AoFeCSO in dealing with ontology object properties

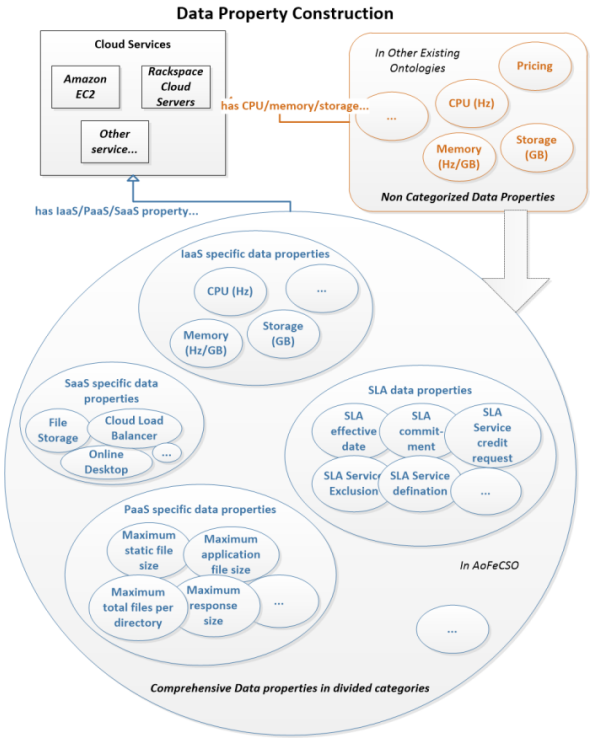

Fig. 3. Advances of AoFeCSO in dealing with ontology data properties

allows DL reasoner to reason new inferred cloud service relationships. In this way, AoFeCSO becomes a densely interconnected ontology model in which very few entities/concepts are seen "isolated" on their own.

3.3.3 Categorized and comprehensive data property assertion

Most of the existing models solely or largely focus on clarifying the numerical data attributes of compute cloud services [10, 11, 12]. In contrast, AoFeCSO employs data properties for much wider specifications. As illustrated in Fig. 3, it employs diverse data types, including String, Boolean, Data time, etc. According to the different cloud services' delivery models, the data properties are divided into a series of sub categories. For instance, IaaS compute services have "vCPU core, frequency, memory size, network performance", etc. PaaS application platform services have "programming language version support, maximum size of application file, maximum total number of file per directory", etc. SaaS file storage services have "binary difference support, file session support, individual size limit, revision history support", etc.

In addition, cloud service SLA data is specified with data property assertions. It involves specifications of SLA descriptions, obligations and other relevant terms and conditions, such as "SLA effective date, service commitment, service compensation, service error rate, service credit request, service annual/monthly up time", etc. [33]. These become a separate complete service data type specification category.

3.3.4 Multi-sourced annotation property assertion

As depicted in Fig. 4, AoFeCSO utilizes annotation properties in a rather different approach against others [8, 34, 35] for entity annotations. It involves annotating not only cloud services, but all other concepts in the ontology, e.g. service models, service characteristics, service properties, CSPs, programming languages, protocols, APIs, etc., regardless of their uniqueness or commonness. In this way, AoFeCSO becomes much more interpretable, even to non-expert users.

Moreover, unlike the existing models which acquire cloud and service (annotation) information from a single knowledge source,

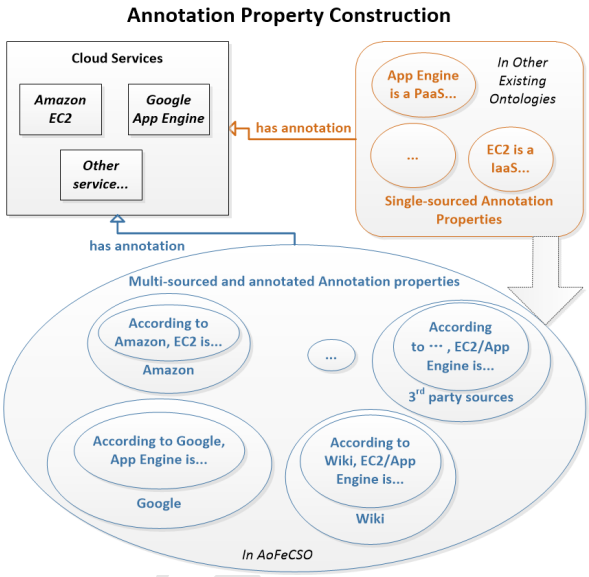

Fig. 4. Advances of AoFeCSO in dealing with ontology annotation properties

AoFeCSO collects and presents multiple descriptions over a diversity of knowledge sources. This establishes trustful entity annotations, since each annotation asserted is accompanied with its origin source information (by annotating the annotation with the source information). Therefore, the multi-sourced annotations enable a more comprehensive view for the target cloud entities.

3.4. Adoption and application of Reasoning OPs

Reasoning OPs are adopted to acquire certain desirable reasoning outcome based on the behavior applied in reasoning engines [36]. Such design enables valuable queries, inferences and ontology evaluation, since it informs the ontology state and allows customized reasoning processes [37]. Typical examples of Reasoning OPs are found as classification, subsumption, inheritance, materialization and de-anonymizing, which are also known as normalizations [38]. Specifically, AoFeCSO adopts Reasoning OPs by applying the following normalization steps.

To present concise class names and eliminate the anonymous class descriptions, many new class names are introduced in AoFeCSO. Complex class descriptions originated from existing ontologies are replaced with ones that make more sense for cloud service entity retrieval tasks. As depicted in Fig. 2, the subclasses of security, scalability and monitoring are new class concepts named from existing knowledge and can be seen as examples of class normalization.

The implementation of the second normalization removes potential anonymous individuals. Basically, each individual entity would own a specific namespace with an URI reference, e.g. cloud services, available VM types/sizes, choices of OSs and software versions bundles, etc. As these entities become unique in AoFeCSO, it enables precise queries while retrieving and comparing information from distinct CSPs.

The subsumption hierarchy materialization and name normalization of AoFeCSO is completed by maintaining only the direct inheritance relationships. Using Protégé [39] and customized DL reasoning inference behavior, this removes the "duplicated" names and axioms originated from other semantic sources while reusing the existing knowledge.

In AoFeCSO, the instantiations of classes and properties are carried out to the deepest level. Evidences can be found in the previous sections and seen from Fig. 2, Fig. 3 and Fig. 4. This

conveys explicit knowledge of cloud service and relevant entities via presenting the granular specifications and details.

Property normalization is done by materializing symmetric and inverse properties, and cleaning the redundant transitive ones where necessary. Several examples of this can be found in AoFeCSO (see Fig. 2), e.g. symmetric properties such as "affects" and "orchestrate with" between cloud service properties, inverse properties such as "controlled by" versus "control over", etc.

As all the above normalizations are applied, AoFeCSO becomes a normalized ontology that is capable of providing diverse cloud service specifications for different (reasoning) needs. Accordingly, the adoption of Reasoning OPs results into considerable valuable queries and new knowledge via ontology reasoning (e.g. entities with similar assertion patterns can be easily categorized, related or differentiated based on customized query or reasoning behaviors).

4. Fuzzy cloud service specification with OWL2 fuzzy extension

To enhance knowledge presentation in terms of capture and revealing the vague/inexplicit cloud service specifications, this section discusses relevant OWL2 fuzzy extension application.

4.1. Fuzziness notation and representation

To explain how the imprecise specifications are implemented in AoFeCSO under PLN theory, we demonstrate some examples. Dropbox [40] is a cloud storage service that allows users to upload, download, synchronize, and share personal files and folders from different OSs/platforms globally. Obviously, service of the kind would own properties such as "reliability"; but "reliability" is a vague term. Further, Dropbox also enables developers to build applications based on the platform; to this extent, it has some PaaS characteristics inexplicitly. While both specifications suffer from the degrees of acceptance (truth) issues, these can be well described according to relevant PLN fuzziness presentations. Specifically, they are known as basic first-order and higher-order logical relationships, which denote (values are example fuzzy data obtained from experiment):

e.g. *IntensionalInheritance Dropbox PaaS* < *[0.3, 0.9] 0.8, 10*>

e.g. *Evaluation hasReliability Dropbox* < *[0.3, 0.9] 0.8, 10*>

The above two statements are to be understood as Dropbox is considered to own PaaS characteristics/reliability attribute at a degree within interval of 0.3 and 0.9 with "creditability" (confidence) of 0.8 and "lookahead" of 10 (i.e. from 10 observations). In contrast to FL representation which can only present a single fuzzy degree value, this comprehensively reveals an interval (as the range of the fuzzy weights), a credibility (of the fuzziness) and the number of evidences (collected from observations).

4.2. Fuzzy data collection

While fuzziness can be very subjective, a closely constructed fuzzy ontology would appear to be subjective, and eventually become unideal. To this extent, we take the initiative to involve users to rate their own perception weights for specification applicability in AoFeCSO. This also complies with the data collection and evaluation processes against relevant PLN theories. By using an integrated user-friendly fuzzy rating mechanism, users do not necessarily require any explicit knowledge of knowledge engineering to make the (rating) contribution. Here, the reputation management framework [41] is adopted for the user expertise classifications. Then, for different user expertise levels, we provide fuzzy rating authorization control for appropriate AoFeCSO input, based on the authorization reference illustrated in Table 1. Indeed, the user expertise profile values obtained from other categorization models can be altered if necessary.

Table 1. Fuzzy Weight Rating Authorization Control

Authority	Beginner	Intermediate	Advanced	Expert
Fuzzy weight update	×	√	√	√
Fuzzy interval update	×	×	√	√
Explicit fuzzy convention	×	×	×	√

Seen in Table 1, the lower the user's level (expertise in cloud computing) is, the smaller the degree of change would be triggered: 1) "Beginners" users are not permitted to input/change any AoFeCSO specifications. 2) Users from "Intermediate" level and up are allowed to donate their own fuzzy ratings according to their understanding for the target specifications. If so, accepted fuzzy rating will trigger a series of ontology update actions, where a new fuzzy value will be recalculated based on the historical rating data stored plus the level of the donating user, under relevant PLN theory. 3) In addition, the fuzzy interval will be updated only if the user is at level of advanced or above. 4) Finally, only "Expert" level users are permitted to make an initial fuzzy rating for a certain specification axiom, as this means to convert a regular axiom from explicit to fuzzy for the first time. The algorithm prevents low level users from making critical changes to AoFeCSO whilst it increases the overall credibility of the applied fuzzy specifications.

4.3. Fuzzy axiom assertion and annotation

To illustrate the transformation of regular to fuzzy ontology assertions plus the impact on the respected ontology reasoning, an example is demonstrated in Fig. 5, using Amazon S3 [42]. Basically, as the service may be considered as SaaS, PaaS or IaaS, three regular delivery model specifications would make no difference among each other ("is delivered as some IaaS/PaaS/SaaS"). In other words, regular assertions can only mean an equal degree of truth among such similar axioms. However, this is inappropriate for most of the cases, as users often find some specifications more applicable than others. Considering S3, the majority agrees that it is more a SaaS than PaaS and IaaS (values are obtained from experiments). With the fuzzy rating information, the fuzzy convention is applied, shown in Protégé snapshots in Fig. 5. The extension is then able to reveal that the "PaaS" delivery model for Amazon S3 is considered to be vague (minority agrees only) with an overall weighted average value of "0.21200001f" ("f" stands for float).

Here, since the fuzzy extension is applied with regular OWL2 data property (with rdfs:Literal schema), after the conversion, the weight-combined axiom becomes an axiom that intersects the original object property assertion and its fuzzy weight data property assertion. This also follows standard OWL2 syntax. As a result, such fuzziness-embedded ontology supports native OWL2 DL reasoner such as FaCT++ [43] and HermiT [44] (see the reasoned/inferred axiom in Fig. 5).

Meanwhile, apart from the fuzzy weight value added onto the original axiom, complete fuzziness data including all historical fuzzy rating information is presented in the annotation field of the fuzzy-extended axiom (see "Annotations" in Fig. 5). With respect to PLN fuzzy data representation, the "Interval" concludes the fuzzy weight interval of the historical rating ranges; the "Credibility" captures the up-to-date credibility of the fuzzy weight ratings; the "Count", which indicates the current total number of ratings, is also known as the "lookahead" value. Additionally, historical detailed rating data for each eligible user expertise level is stored, which comprises the average values and counts for "Intermediate", "Advanced" and "Expert" users respectively.

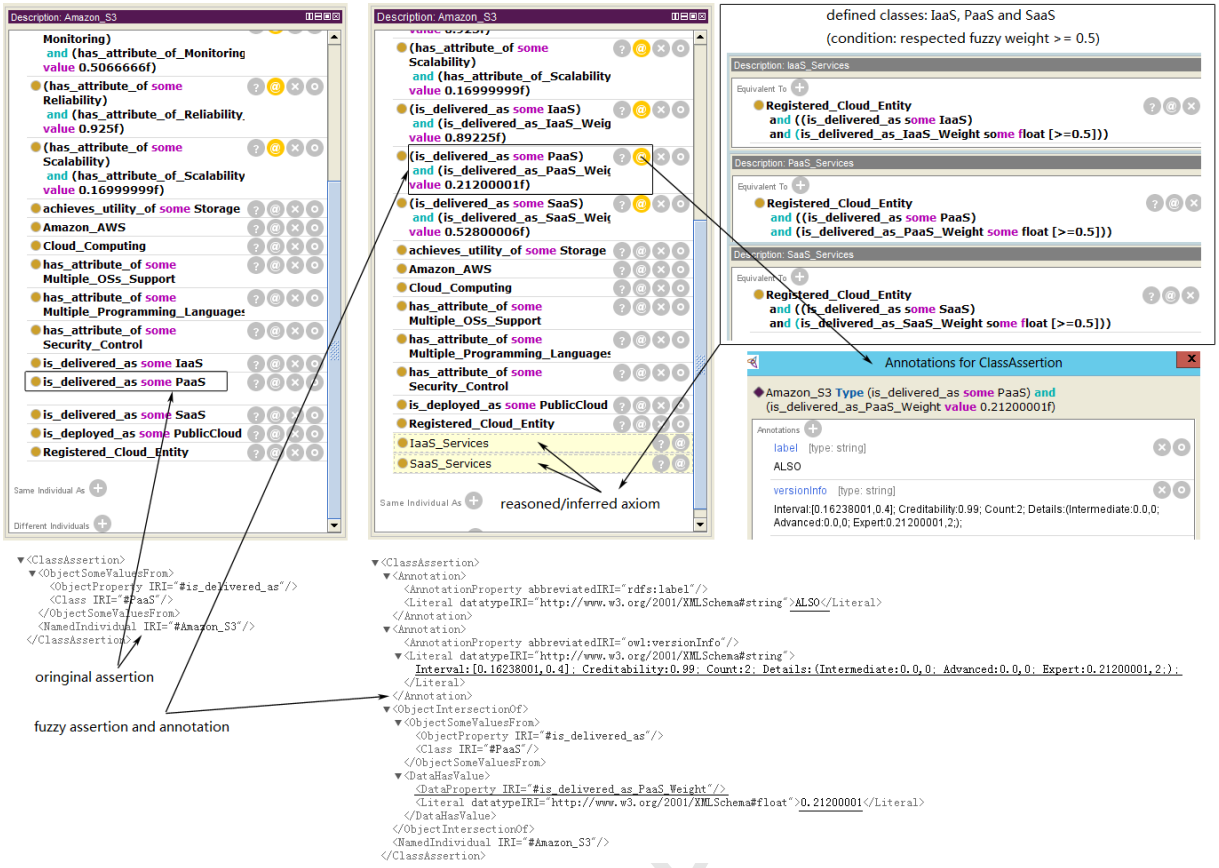

Fig. 5. Fuzzy conversion, annotation and reasoning in AoFeCSO

Let FW represents fuzzy weight, $C_{overall}$ represents the overall credibility, the equations for fuzzy weight and credibility calculation take the form:

$$FW = \frac{\overline{R_I}*C_I*N_I+\overline{R_A}*C_A*N_A+\overline{R_E}*C_E*N_E}{C_I*N_I+C_A*N_A+C_E*N_E} \quad (1)$$

$$C_{overall} = \frac{C_I*N_I+C_A*N_A+C_E*N_E}{N_I+N_A+N_E} \quad (2)$$

where

$$\overline{R_I} = \frac{\sum_{i=1}^{N_I} R_{Ii}}{N_I}$$

$$\overline{R_A} = \frac{\sum_{i=1}^{N_A} R_{Ai}}{N_A} \quad (3)$$

$$\overline{R_E} = \frac{\sum_{i=1}^{N_E} R_{Ei}}{N_E}$$

Here $\overline{R_I}$, $\overline{R_A}$, and $\overline{R_E}$ represent the average rating values of "Intermediate", "Advanced" and "Expert" users respectively, for each ratings R_{Ii}, R_{Ai} and R_{Ei}; C_I, C_A and C_E represent the credibility values of each respected user levels; N_I, N_A and N_E represent the number of total ratings of the different user levels. From the equations, it can be seen that whenever a new rating is accepted, the fuzzy weight and overall credibility is recalculated whilst a series of detailed data fields are updated.

4.4. Fuzzy axiom management

The process of ontology fuzzy modification is described as follows. When a new fuzzy rating is detected, it is first verified against the authorization control specified in Section 4.2. Afterwards, in case of an initial fuzzy weight assertion (explicit-to-fuzzy conversion), a series of fuzziness statements and parameters are created in the format illustrated in Section 4.3 at first. Due to the fact that it is the first rating, the credibility would be 100% whilst the interval is set to +/-10% of the rating value. Followed by that, an ad-hoc data property is created using a name which combines the name of the object property and class plus the word "Weight", indicating this is a specific restriction applied onto the target axiom. The value of the ad-hoc data property, also known as the fuzzy weight, is simply the rating entered by the expert user.

For fuzzy weight update, the existing fuzziness data is retrieved and validated at first. Then, based on the new rating, appropriate fields are updated according to (3). As the updates complete, a new fuzzy weight and the overall credibility value are recalculated using (1) and (2).

While all fields of the detailed fuzziness data and fuzzy axioms are successfully created/updated, a fuzzy annotation label is also prepared for the fuzzy axiom, based on the new fuzzy parameters as well as the nature of the axiom: e.g. with a weight of (0,0.5)/[0.5,1), "STRONG/WEAK" on a service property axiom suggests that the cloud service is strongly/weakly considered to own the property; "DIRECT/INDIRECT" for a service functionality axiom implies such is a primary/secondary function of the service; "MAIN/ALSO" over other assertions state that the assertions are mainly/also argued as such. These further explanations help users better understand the fuzzy weight values with respect to the nature of the information they reveal.

Next, all above updated contents are imported to a temporary ontology where the relevant contents are modified. If there is no error occurred after the updates, the reasoning process will be initiated to check for any inconsistency or new inferred axioms. Here, any new inferred axioms, if it exists, will also be saved to the ontology, whereas the original ontology data will be restored

if there is any updating/saving error occurred or inconsistency detected.

Additionally, all the fuzzy extensions (fuzzy weight axioms, rating information annotation) can be eliminated, if some prefer a typical/traditional cloud service ontology (with agility-centric design). This is done by removing the fuzziness data properties plus the relevant fuzzy annotations. As the fuzziness axioms are not closely related with the core ontology architecture, this will not affect the original explicit knowledge presentation.

5. Prototype implementation

CSR prototype is implemented in Java. As depicted in Fig. 6, it comprises Active Ontology Manager, Authorization Manager, Service Search & Recommendation Engine and User Interface four main components.

5.1. System components

User Accounts and Profiles Database stores user account data, which is used for Authorization Manager to authorize actions such as service information access, recommendation and fuzzy rating actions. Basically, all users can access the service specifications via Service Seeker, Service Explorer and Service Recommender; yet for inputting fuzzy ratings, restricted controls are applied according to users' expertise levels (based on Table1).

Service Search and Recommendation Engine takes input of both user's preference entries and their profiles to provide service search and recommendation functions. Through pre-set SPARQL query clauses and API queries, service discovery is implemented by collecting services for keyword/filter matches; service recommendation is performed by evaluating services' specifications against user weighted importance factors.

User Interface consists of Account Manager, Service Explorer, Service Recommender, and Service Seeker interfaces. Account Manager allows users to fill in and edit their account and profile details. Service Seeker provides flexible service search and filter options. Service Recommender produces service lists and recommendation ratios based on user-defined recommendation conditions. Service Explorer presents service specifications through a number of tabs, i.e. General Description, General Attributes, Detailed Attributes, and Agility Breakdown.

Active Ontology Manager manages AoFeCSO through OWL API [45]. It incorporates Entity and Axiom Manager, Ontology Reasoning Manager, Ontology Evolution Engine, and Revision and Rollback Manager four subcomponents. Entity and Axiom Manager interprets the ontology axioms whilst it makes changes to them according to certain user requests. It deals with both regular and fuzzy ontology specification interpretation and modification tasks. Ontology Reasoning Manager handles ontology consistency checks and inference controls through binding OWL2 reasoner. The reasoner adopted here is FaCT++, due to its faster response plus better syntax and property characteristics support [43]. In case of ontology specification modification, a temporary ontology copy will be created at first, whereas Ontology Evolution Engine will attempt to discover new knowledge through reasoning inference automatically: as the reasoning process is complete, the consistent temporary ontology plus any new inferred axioms (specifications) will be saved and then replace the active ontology. This is how AoFeCSO evolves progressively while remaining absolute consistency. Revision and Rollback Manager maintains and conserves redundant ontology copies, i.e. Historical Ontology Copies. This enables ontology recovery in case of failures occurred during modification.

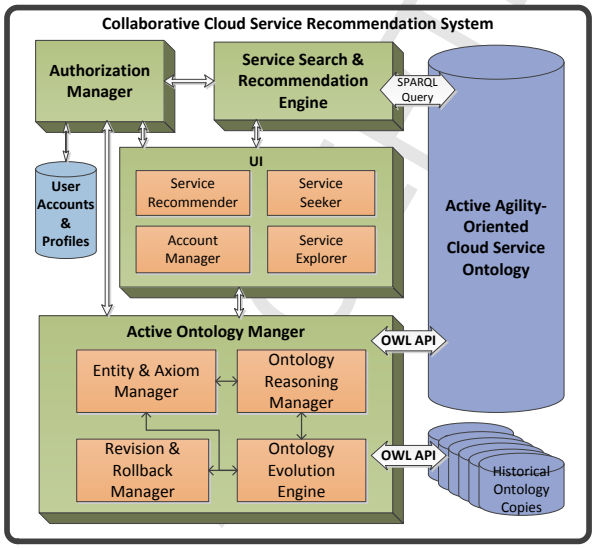

Fig. 6. CSR system architecture

5.2. Service profile (agility) evaluation

The evaluation of a cloud service's agility is based on all the specifications that are relevant to the service. An agility score is calculated according to three evaluation criteria. Let PA, SA and TA represent primary, secondary and tertiary agility aspects, the assessing equation takes the form:

$$\text{AgilityScore} = \text{PA} + \text{FW}_{\text{SA}} * \sum_{\text{I}=1}^{\text{N}_\text{I}} \text{SA}_\text{I} + \sum_{\text{i}=1}^{\text{n}_\text{i}} \text{TA}_\text{i} \quad (4)$$

where N_I and n_i are the total numbers of the secondary and tertiary aspects found, FW_{SA} is the asserted fuzzy weight of the aspect.

Basically, primary agility criterion accounts for 50% of a service's agility score, which is determined by the service's function utilities (e.g. resource/platform/software provisions, etc.). Secondary agility criterion takes 40% of the total agility score, which is decided based on the service's main characteristics and features (e.g. scalability, elasticity, API, OS/programming language support, etc.). Tertiary agility criterion makes up the rest 10%. It tracks the total number of other service attributes that are regarded weakly relevant to agility (e.g. logging access, application deployment support, migration and transition support, customer service and negotiation support, etc.).

5.3. Service recommendation

Cloud service recommendation is implemented based on user selected weighted recommendation keywords. The process starts by asking for relevant information (keywords) for the target cloud services. The keywords can be of any categories, e.g. services' functions, features, characteristics, etc. The selectable keywords are arranged in a hierarchical layout according to relevant structure/relationships defined in AoFeCSO. Further, to assist users in understanding the unfamiliar terminology, multi-sourced annotation explanations of the keywords are retrieved and displayed.

During the selection process, users can specify the degrees of importance for each keyword selected. With the list of the weighted recommendation keywords, the recommendation engine scans AoFeCSO and analyzes all the specifications for each candidate cloud service. Then, for the services which comply with the keywords, recommendation ratios are calculated and displayed:

$$\text{Ratio}(\text{Service}_\text{n}) = \frac{\sum_{\text{I}=1}^{\text{N}_\text{I}} \text{I}_{\text{K}_\text{I}} * \sum_{\text{i}=1}^{\text{n}_\text{i}} \text{I}_{\text{ki}} * \text{FW}_\text{i}}{\text{n}_\text{i}} \quad (5)$$

where I_{K_i} is the main importance degree of the home service keywords category, N_i is the number of the home categories selected, I_{ki} is the sub importance degree of the sub service keywords, FW_i is the fuzzy weight of the encountered service specification if applicable, n_i is the total number of the sub keywords selected for recommendation.

Finally, a recommendation result is produced. It contains a list of cloud services which are accompanied by certain computed recommendation ratios. The ratios indicate the applicability that the recommended cloud service would fit for the specified weighted service requirements.

5.4. Component interactions

The main interactions among the above system components are seen as follows. Basically, any ontology modification requests must go through authorization checks at first. Ontology Reasoning Manager is called every time AoFeCSO is successfully updated, either by Entity and Axiom Manager (due to new information added) or Ontology Evolution Engine (due to any new ontology copy saved). Then, 1) if the temporary ontology is inconsistent, it will notify Entity and Axiom Manager to discard the temporary ontology and changes and tell the users the inconsistency along with the cause; 2) if the temporary ontology is consistent and free from new inferred knowledge, it will be forwarded to Ontology Evolution Engine where it will be deployed and take place of the current live ontology; 3) if the temporary ontology is consistent with the updates whilst there are new inferred axioms, the details will be sent back to Entity and Axiom Manager to notify the system user, where upon acceptance the temporary ontology along with the inferred axioms will be saved. Revision and Rollback Manager only receives calls from Ontology Evolution Engine when it fails to deploy the new ontology with the updates. Furthermore, the system components are controlled with a deadlock and queuing mechanism, which prevents possible concurrent actions during the ontology modification, temporary ontology creation, reasoning processes, and ontology replacement processes.

6. Case study

To illustrate how AoFeCSO captures cloud service specifications and how the information can be interpreted for cloud service search, recommendation and retrieval tasks, we provide a case study using Google AppEngine [46] (see Fig. 8).

6.1. Cloud service search with keywords and filters

With the stored service specifications, the search functions are provided with two main processes: keyword-based search and restriction filter. The former attempts to find any cloud services which are relevant to the entered service information. The latter seeks services which fulfill the applied restriction information. Here, in case of multiple keywords, any services with at least one (word) match would be selected; if multiple restriction filters are used, only the services which satisfy all the filters would be selected. The two processes can be used together or separately.

Seen the example search in Fig. 7, as a user enters “PaaS, elasticity, database”, etc. words, the search would output all cloud services which are specified as PaaS, or with elasticity, or directly/indirectly offers database functions, etc., from applicable CSPs. Then, as a series of filters are deployed, the service search result lists are reduced based on whether they would fit into the restrictions. Users can freely use the given filter terms (which are acquired from AoFeCSO), or insert customize restrictions using texts, numerical values and symbols. As a result, the proposed approach enables much more flexible cloud service search.

6.2. Cloud service recommendation with ratios

The recommendation result demonstrated in Fig. 7 is obtained from a series of weighted service information keywords (displayed at the top of the “recommendation” panel).

The ratios next to the recommended services reveal how well the candidates fit into the recommendation profile. The example demonstrates that PaaS services like IBM SmartCloud [47] and AppEngine have the highest applicability for those selected weighted keywords. As a user selects a service, the relevant service specifications will be displayed on the right, indicating the details of the ratio constitution.

6.3. Cloud service retrieval

Fig. 8 shows examples of CSR (Service Explorer tabs) displaying AppEngine specifications. The specifications are dynamically retrieved from AoFeCSO and are arranged into a series of categories.

(1) Cloud service descriptions, which are modelled with entity annotation assertions, are displayed on the “General Description” tab. For the AppEngine example, there are two descriptions originated from two sources: “Wiki” and “Official”.

(2) General cloud service attributes, which are modelled with entity superclass assertions, are interpreted in the “General Attributes” tab. This often involves the service delivery model, deployment type, function, feature, etc. specifications. For instance, AppEngine belongs to “PaaS” and “Public Cloud”; it has functions of “Application Development & Testing”; it has features of “Adaptability” and “Application Development Support”. Further, for specifications which are considered to be

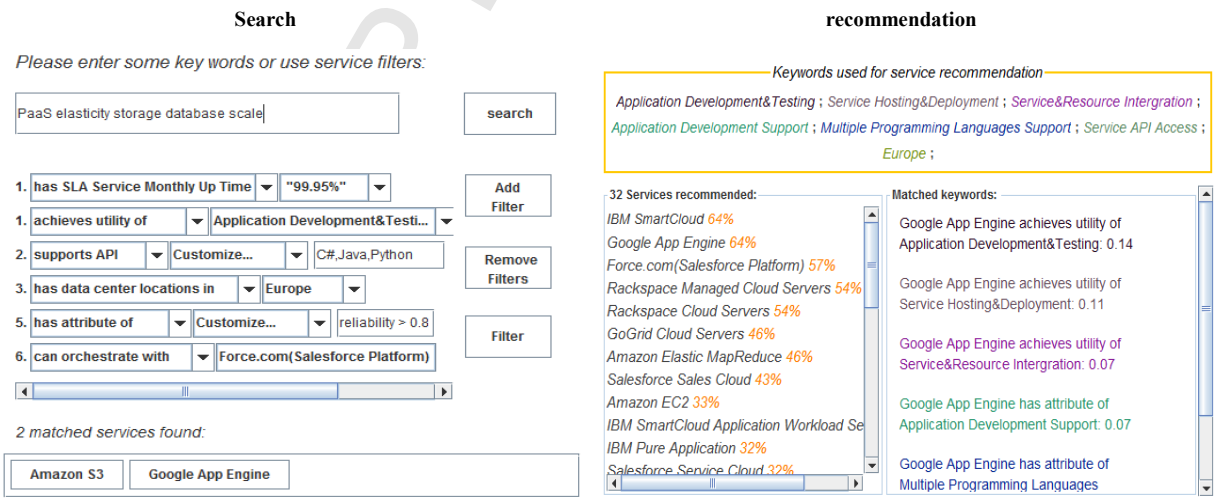

Fig. 7. CSR Screenshots for cloud service search and recommendation

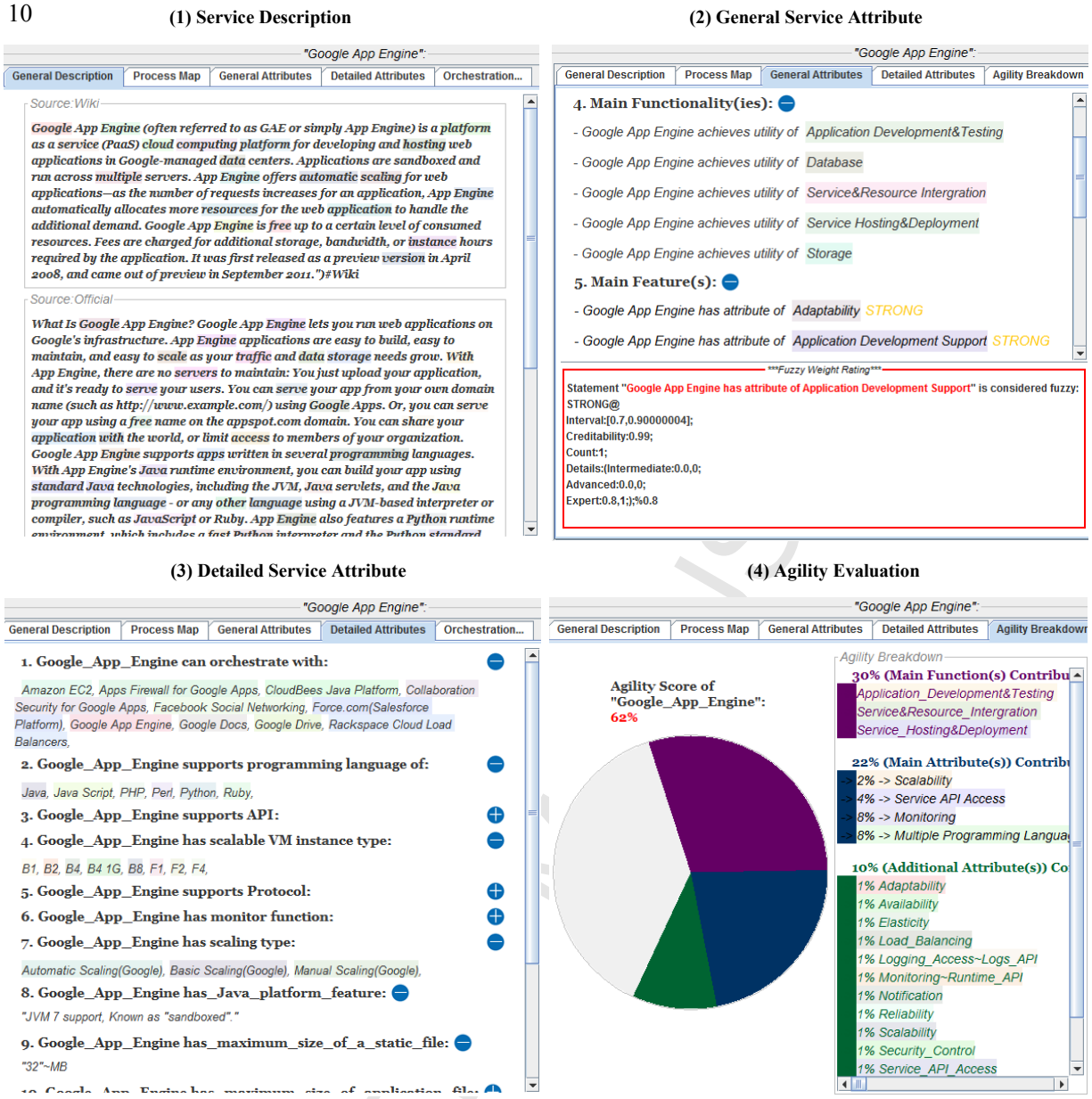

Fig. 8. CSR Screenshots for cloud service retrieval (Google App Engine)

inexplicit, users can view/edit their truth degrees. The "STRONG" and the example fuzzy weight information suggest that AppEngine offers good "Application Development Support" features.

(3) For any details regarding cloud service general attributes or specific specifications, they are to be found on the "Detailed Attribute" tab. For instance, AppEngine is capable of orchestrating with other cloud services such as "Amazon EC2", "CloudBee Java Platform", "Google Drive", etc. It supports multiple programming languages including "Java", "PHP", "Python", etc. It has scalable VM instance type of "B1", "B2", "F4", etc. whilst it offers scaling type of "Automatic Scaling", "Basic Scaling" and "Manual Scaling". It supports Java platform feature of "JVM 7 (sandboxed)". It has a maximum static file size limit of "32MB". Such data is modelled with the individual's object property and data property assertion in AoFeCSO, which guarantees the accuracy of the specification semantics and knowledge presentation.

(4) Finally, all of the service specifications are evaluated, which are then used to produce its overall agility score. The score plus relevant agility constitutions are summarized in the "Agility Breakdown" tab. Consequently, the series of dynamically retrieved and arranged service specifications suggest that the proposed approach can adequately capture and present a variety of cloud service information for AppEngine.

7. Evaluation and discussion

For evaluation, we discuss a series of aspects according to state-of-the-art ontology evaluation approaches [48, 49].

7.1. Domain coverage

In ontology evaluation, domain coverage attempts to justify the ontology knowledge coverage in contrast with other modelling practices (e.g. existing gold standard ontologies, other model sources, etc.) [49]. Here, we compare AoFeCSO with a number of existing cloud (service) ontologies in terms of both the coverage scale and the details.

Table 2 summarized the domain coverage scales of existing cloud (service) ontologies. Indeed, most of the existing ontologies often concentrate on specific service delivery models.

Table 2. Domain Coverage Scale

Coverage	Cloud ontologies	mOSAIC [58]	Unified business and cloud service ontology[9]	FCFA [10]	CoCoOn [11]	Cloud ontology [12]	Cloud Ontology [32]	Business ontology [56]	AoFe-CSO
IaaS cloud entities and properties	Compute	√	√	√	√	√	√		√
	Network	√	√	√	√	√	√		√
	Storage	√	√	√	√	√	√		√
PaaS cloud entities and properties	Application development & testing	√	√			√	√	√	√
	Application deployment & hosting	√	√			√	√	√	√
	Service & resource integration	√	√						√
SaaS cloud entities and properties	Business process & intelligence	√	√				√	√	√
	Cloud & web resource management	√		√					√
	General software application		√			√	√	√	√
Other cloud entities and properties	Party/actor/role	√							√
	SLA/contract	√		√				√	√
	Billing		√		√	√	√	√	√
	Requirement	√						√	

Table 3. Details of AoFeCSO

No. of classes	No. of individuals	No. of object properties	No. of data properties	No. of annotations	No. of axioms	DL expressivity
1231	913	134	537	2544	27932	*SROIF(D)*

Table 4. Service attributes processing effectiveness: service recommendations

Cloud service recommendations	Other existing practices	AoFeCSO & CSR
Description/ explanation of the keywords	Few, partially, single source [4, 8, 35]	Full, multiple sources
Cross/multiple service categories/models	Partial [4, 32]	Yes
Fuzzy cloud specifications considered	N/A	Yes; processed during the recommendation process and represented in the recommendation ratios

Table 5. Overall service attributes processing effectiveness

Overall effectiveness comparison	Other models and service recommendation systems	AoFeCSO & CSR
Description of service attribute	Yes [4, 8, 9, 11, 12, 34, 32, 56, 57]	Yes
Granular service attribute details	Very few [4]	Yes
Service attribute connections	N/A	Yes
Service attribute fuzziness specification	N/A	Yes, through collaborative fuzzy weight rating
Service/provider relationships	N/A	Yes

Hence, they would present only partial knowledge of certain service categories. Only AoFeCSO and mOSAIC cover the entire cloud service models. The main differences between the two ontologies are seen twofold: 1) AoFeCSO does not involve any CSC requirement aspects whilst mOSAIC does not provide cloud service billing specifications. 2) AoFeCSO provides focus-neutral specifications and would not over-concentrate on any specific cloud service models for details; in contrast, mOSAIC lacks some SaaS descriptions. Accordingly, these suggest that the proposed ontology owns a competent domain coverage.

7.2. Quality of modelling

Ontology modelling quality is often assessed based on its syntactic, structural and semantic quality aspects [50], where the logical consistency must be guaranteed. AoFeCSO is (initially) built using Protégé. This means that it follows formal OWL2 syntactic features for axiom assertions. Table 3 describes the details of AoFeCSO in terms of the total numbers of classes, individuals, object properties, data properties, annotations, axioms, plus its DL expressivity. As an active ontology, its DL consistency has been automatically verified (by FaCT++) whenever any new information is added.

AoFeCSO adopts Reasoning OPs. It has been kept to the series of ontology normalization processes through the construction cycle. While this not only guarantees the standard and quality of the ontology, it also drives the desired reasoning outcome, e.g. inferred cloud (service) entity specifications such as inferred membership functions, property constraints and other object relationships.

7.3. Suitability for service retrieval and recommendation tasks

For the suitability evaluation, we compare AoFeCSO with other existing service specification models for service retrieval and recommendation tasks.

Regarding the suitability of the service recommendation tasks, the proposed approach is found to be advanced in three main aspects (refer to Table 4): I) It facilitates a user-friendly recommendation process due to the comprehensive keywords annotation presentation, whilst this assistance feature is seldom available in other cloud service recommendation tools. II) It is by far the first tool that provides comprehensive service recommendation functions for diverse service models and categories. III) The recommendation functions consider the fuzziness occurred in cloud service specifications; this enables a clearer view of the small differences between similar cloud services through more precise service recommendation ratios.

Additionally, as Table 5 summarizes, the proposed approach is able to capture and present extended service specifications from a variety of aspects, e.g. to show multiple service model information, explaining granular details of service attributes, revealing service attribute connections, and processing fuzzy service specifications. Fundamentally, we argue that other existing work is held back by their conventional inflexible model definition and implementation, whereas our approach rests on a loosely-coupled class and relation hierarchy.

Concluded from the above case study and comparison data, AoFeCSO and CSR offer distinguished effectiveness for cloud specification processing with regard to the full range service recommendation and retrieval tasks.

7.4. Adoption and use

In addition to the present use, AoFeCSO is also actively involved in a number of research projects. Indeed, its knowledge is being widely used for recent cloud service orchestration [51] and brokerage [52] studies. While being adopted to assist service optimization tasks, it can provide adequate semantic support to compare cloud services with similar functions, features, characteristics, etc. Further, as being used for service brokerage tasks, it would greatly enhance service matchmaking for cloud (resource) interoperability enablement. Indeed, the comprehensive service specifications across multiple abstraction layers make it a preferred knowledge for a wide range of service selection-relevant tasks.

8. Related work

8.1. Ontology-based knowledge representation on web services and cloud

In the last decades, XML-based modelling specifications have been widely utilized in semantic web services. Indeed, while describing service resources, functions, properties, etc., the RDF vocabulary and syntax provide a structured data presentation which can be effectively interpreted and processed. This enables and enhances relevant service discovery, selection, matchmaking, and composition tasks for a variety of purposes [51, 54]. OWL, as an advanced semantic modelling language from this origin, offers even more features mainly due to its further DL reasoning capabilities [6].

Particularly, research on cloud computing/service semantic modeling involves various ontological approaches, such as single ontology [9], multiple-layered ontologies [7] and multiple ontologies [10], etc. The semantic platform for cloud service annotation and retrieval [8] utilizes multiple ontologies of different domains. Being advanced in its annotation term extraction and indexing techniques plus the integrated ontology evolution module, it can implement ontology updates according to the service concept information found on Wikipedia. In their incremental work [34], GATE [55] is employed for automatic service annotation and ontology evolution. Nonetheless, a limitation is that annotation specification update does not affect the ontology structure or any other assertions, as they do not participate in ontology reasoning process.

Alternatively, other work (e.g. [11, 56]) employs class, object property and data property assertions with relevant DL rules and inference in the proposed ontologies. Nevertheless, most of the models are primarily designed to work for certain limited service categories: e.g. infrastructure services [10, 11, 12, 32], platform services [9, 56] and software services [8, 34]. FCFA [10], for instance, is a hierarchical federated resource exploration and sharing framework model which drives federated cloud cooperation and eliminates interoperability issues among independent organizations and providers. The model only concentrates on the relationships between organizations and communities in terms of federation contracts, SLA agreements, the various physical and virtual resource properties, etc. CoCoOn [11] is an infrastructure service model which comprises both functional and non-functional specifications of cloud virtual machine (VM) and storage resource aspects; it still does not involve service information across wider resource abstraction levels. Although Cloud Ontology [12] is able to specify service information of a variety of cloud services, it only discloses some basic aspects regarding the diverse service functions and levels. In fact, for the existing models, the cloud computing/service concept specifications are seldom established evenly across multiple abstraction levels and service function categories. Indeed, except mosaic [4], very few ontologies touch the explicit details of both functional and non-functional properties of diverse cloud services types. Besides, no other ontology attempts to specify the several service agility aspects or the most appropriate specifications through fuzzy extensions; none of the current practices supports collaborative model editing for the field.

8.2. Cloud service recommendation systems

Existing service recommendation/discovery systems/tools are seen limited in terms of their overall applicability, flexibility and comprehensiveness. Some [11, 32] are found focusing on IaaS-centric service recommendation. Specifically, CSDS [32] presents a discovery system for VM services according to search parameters such as virtual CPU architecture/frequency, memory/storage size, network parameter, operating system (OS), etc. CloudRecommender [11] offers enhanced functions which accept both functional and non-functional service properties as recommendation requirements. Nonetheless, due to their limited service category applicability, the two systems cannot facilitate comprehensive service recommendation in a wider domain (e.g. PaaS and SaaS). Differently, the cloud repository and discovery framework [9] advocates a cloud service recommendation approach based on a business and cloud service combined ontology. However, since the recommendation is implemented through querying business-relevant service properties, it implies that the recommendation process would be excessively business–focused. Cloudle [12] can produce a list of discovered services along with their similarity values from several services types by offering diverse search criteria and options such as cost, time, function, technical requirements, etc. Yet, the similarity computation relies on purely numerical service properties and, therefore it still cannot effectively handle comprehensive service specification. On the other hand, non-ontology-based service recommendation system, like the collaborative service recommender mechanism [57], is an alternative that specifically deals with service matchmaking through consumer rated service qualities against users' profiles. Yet due to the prototype mostly concentrated on non-functional service aspects (e.g. response time, availability, price, etc.), the limited functional requirement processing capability would result into poor overall service recommendation.

Indeed, currently there is not a comprehensive means of cloud service search, retrieval and recommendation which covers a diversity of service/application domains, whereas none existing tool attempts to involve search/recommendation requirements regarding any details regarding the unique (agility) aspects of cloud services, e.g. scalability, adaptability, interoperability, etc.

8.3. Ontology fuzzy extensions

On the basis of the FL theories, a series of OWL/OWL2 fuzzy extension techniques propagate. FuzzyOWL2Ontology [59] advocates a merging approach to import the fuzzy representations, which are wrapped as ontology entities, to the target ontology for fuzziness expression. The drawback is the limited support of complicated fuzzy scenarios plus the considerable extra overhead. In contrast, new syntax-based fuzzy extension [60] is proposed where the primitive OWL2 syntax is extended with "owlx:degree", "owlx:ineqType", etc. elements.

Nevertheless, without specific extension mechanism/plug-in support (for fuzzy assertion and interpretation), the modification has little compatibility with main stream OWL/OWL2 tools. The annotation-based fuzzy extension [21] presents another approach, seen as to place the fuzziness in OWL2 annotations. With comprehensive fuzzy set and relation theory support using "fowl" and "fuzzyOWL2" syntax, a Protégé plug-in is developed for easy fuzzy modification and illustration. Yet, fuzzy annotations would only provide entity fuzzy descriptions whilst they do not influence any other property assertions in the ontology.

As all the above approaches remain unideal, the OWL2 natively supported fuzzy extension [22] demonstrates a promising technique by using fuzzy tag-alike modifications. The extension employs no further new syntax but only OWL2 data property assertions. This brings a series of advantages: the fuzzy extended ontology is interpretable by all mainstream OWL2 tools and reasoning engines; the asserted fuzziness can trigger ontology (reasoning) inference changes. Due to the advances, we advocate its extended version along with relevant PLN theory support for ontology fuzzy specification in AoFeCSO.

To summarize, existing cloud computing/service semantic models are often based on unbalanced and incomprehensive service and concept specification establishment. For most of them, explicit details regarding services' characteristics, properties and relationships are missing. Moreover, no existing model involves the specification and presentation of cloud service fuzziness. Consequently, they have limitations in terms of the comprehensiveness and depth of the knowledge represented; particularly, they fail to deal with service agility across the abstraction levels and the service categories. These issues prevent current service recommendation systems from providing the most effective cloud service recommendation functions. In fact, fundamentally, this is very likely caused by the conventional inflexible design accompanied by the DL-consistent nature of OWL ontology. From a range of proposed FL-based ontology fuzzy extensions, we adopt the new PLN-based OWL2 natively supported fuzzy extension to develop the loosely-coupled and agility-oriented cloud service model and the resultant service recommendation system. As such fuzziness is imported in a collaborative manner (via fuzzy ratings), the proposed approach ought to achieve an ultimate cloud service semantic model towards comprehensive and flexible service search, retrieval and recommendation.

9. Conclusions and future work

The continuously propagated cloud services have imposed strong requirements for cloud service specification models and service recommendation systems. Meanwhile, existing cloud computing/service models cannot cover the diverse cloud service concepts and their interactions across different function categories and abstraction levels. The existing cloud service recommendation tools would not handle the unique cloud service characteristics, properties and orchestrations.

This paper presents a novel cloud service semantic model named AoFeCSO. It adopts loosely-coupled, agility-oriented and fuzziness-embedded deployment by introducing multiple sourced annotation assertions, functionally categorized data property assertions, and explicit cloud service concept relations. Additionally, in contrast with existing models which are managed exclusively and statically, AoFeCSO is maintained collaboratively and can evolve accordingly. Users can not only explore the model, but also contribute their own knowledge to it interactively. This significantly enhances the specification and presentation of cloud service information.

A prototype CSR tool is developed on top of AoFeCSO for collaborative service search, retrieval and recommendation tasks. The case studies and evaluation suggest that the model and tool can overcome various existing limitations with effective service explore and recommendation assistances. Although currently AoFeCSO has not many use and reuse applications, this is mainly due to its short establishment.

The future work will target at extending the proposed model by enabling further model collaboration and evolution , e.g. to allow CSPs to add services, change service specifications, etc.; to allow CSBs to specify service interactions and orchestrations, etc.; to allow CSCs to complete service usability ratings, reviews, etc. An open fuzzy specification handling API is to be provided to assist the collaboration. We believe this collaborative manner of cloud service model specification, maintenance and update to be a distinguished means in providing knowledge sources for ultimate service search, retrieval and recommendation tasks. Further, the prototype tool can be enhanced with some user-centric functions. This can be achieved by meeting the specific needs and knowledge based on different user expertise levels.

Acknowledgements

We acknowledge the support from a joint grant by the British Royal Society and the Royal Irish Academy on a Cloud Migration Framework 2014-2016, IE131105.

References

1 D.C. Marinescu, Cloud computing: Theory and practice, Elsevier, Waltham, USA, pp. 2-17, 2013.
2 R.Buyya, C. Vecchiola and S. Thamarai, Master cloud computing: Foundations and applications programming, Elsevier, Waltham, USA, pp. 3-27, 2013.
3 OASIS, TOSCA overview, available at: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca#overview, 2014.
4 R. Aversa, B.D. Martino, F. Moscato, D.Petcu, M. Rak and S. Venticinque, An ontology for the cloud in mOSAIC, Cloud Computing Book 2010: Cloud Computing: Methodology, System, and Applications, CRC Press, pp. 467-486, 2011.
5 HEADS, HEADS project overview, available at: http://heads-project.eu/objectives, 2014.
6 J.M.S. Orozco, Applied ontology engineering in cloud services, networks, and management systems, Springer Science+Business Media, pp. 23-52, 2012.
7 J. Shen, G. Beydoun, G. Low and L. Wang, Aligning ontology-based development with service oriented systems, Future Generation Computer Systems, vol. 32, pp. 263-273, 2014.
8 M.A. Rodríguez-García, R. Valencia-García, F. García-Sánchez and J. J. Samper-Zapater, Creating a semantically-enhanced cloud services environment through ontology evolution, Future Generation Computer Systems, vol. 32, pp. 295-306, 2014.
9 A. Tahamtan, S.A. Beheshti, A. Anjomshoaa and A.M. Tjoa, A Cloud Repository and Discovery Framework Based on a Unified Business and Cloud Service Ontology IEEE World Congress on Services (SERVICES), pp.203-210, 2012.
10 G. Manno, W.W. Smari and L. Spalazzi, FCFA: A semantic-based federated cloud framework architecture, International Conf. High Performance Computing and Simulation (HPCS), pp.42-52, 2012.
11 M. Zhang, R. Ranjan, A. Haller, D. Georgakopoulos, M. Menzel and S. Nepal, "An ontology-based system for Cloud infrastructure services' discovery, 8th International Conf. Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), pp.524-530, 2012.
12 J. Kang and K. Sim, Cloudle: A Multi-criteria Cloud Service Search Engine, IEEE Asia-Pacific Services Computing Conference (APSCC), pp.339-346, 2010.
13 M.K. Smith, C.Welth and D.L. McGuinness, OWL Web Ontology Language Guide, W3C Recommendation, available at http://www.w3.org/TR/owl-guide/, Feb. 2004.
14 H. Nacer and D. Aissani, Semantic web services: Standards, applications, challenges and solutions, Journal of Network and Computer Applications, vol. 44, pp. 134-151, 2014.
15 A. Tahir, D. Tosi and S. Morasca, A systematic review on the functional testing of semantic web services, Journal of Systems and Software, vol. 86, no. 11, pp. 2877-2889, 2013.
16 L. Li, D. Liu and A. Bouguettaya, Semantic based aspect-oriented programming for context-aware Web service composition, Information Systems, vol. 36, no. 3, pp. 551-564, 2011.
17 C. Ke and Z. Huang, Self-adaptive semantic web service matching method, Knowledge-Based Systems, vol. 35, pp. 41-48, 2012.
18 J.J. Jung, Semantic business process integration based on ontology alignment, Expert Systems with Applications, vol. 36, no. 8, pp. 11013-11020, 2009.
19 A.L.V. Antwerp, K. Scoboria and J.R. Santos, Security guidance for critical areas of focus in cloud computing v3.0, Cloud Security Alliance, available at: https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf, 2013.
20 W. Jansen and T. Grance, Guidelines on security and privacy in public cloud computing, Draft NIST Special Publication, 2013, available at: https://downloads.cloudsecurityalliance.org/initiatives/ guidance/NIST-Draft-SP-800-144_cloud-computing.pdf, 2013.
21 F. Bobillo, U. Straccia, Fuzzy ontology representation using OWL2, International J. Approximate Reasoning, vol. 52, no. 7, pp. 1073-1094, 2011.
22 D. Fang, X. Liu, I. Romdhani and H. Zhao, Towards OWL 2 Natively Supported Fuzzy Cloud Ontology, 36th Annual Computer Software and Applications Conference Workshops (COMPSACW), pp. 328-333, 2012.

23 L.A. Zedeh, Fuzzy set, Information and Control, vol. 8, pp. 338-353, 1965.

24 T. J. Ross, Fuzzy logic with engineering applications, Third Edition, John Wiley & Sons, Ltd, Chester, UK, pp 48-73, 2010.

25 B. Goertzel, M. Iklé, I.F. Goertzel and A. Heljakka, Probabilistic Logic Networks - A Comprehensive Framework for Uncertain Inference, Springer, pp.1-148, 2008.

26 K. Jeffery and B. Neidecker-Lutz, The Future of Cloud Computing: Opportunities for European cloud computing beyond 2010, available at: http://cordis.europa.eu/fp7/ict/ssai/docs/cloud-report-final.pdf, 2013.

27 V. Hahn, L.JR. Santos, K.Scoboria, E.Scoboria and J.Yeoh, Secaas implementation guidance: Web security, Cloud Security Alliance, 2013, available at: https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_3_Web_Security_Implementation_Guidance.pdf, 2013.

28 V. Hahn, L.JR. Santos, K.Scoboria, E.Scoboria and J.Yeoh, Secaas implementation guidance: Security assessments, Cloud Security Alliance, 2013, available at: https://downloads.cloudsecurity alliance.org/initiatives/secaas/SecaaS_Cat_5_Security_Assessments_Implementation_Guidance.pdf, 2013.

29 V. Hahn, L.JR. Santos, K.Scoboria, E.Scoboria and J.Yeoh, Secaas implementation guidance: Encryption, Cloud Security Alliance, 2013, available at: https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_8_Encryption_Implementation_Guidance.pdf, 2013.

30 V. Hahn, L.JR. Santos, K.Scoboria, E.Scoboria and J.Yeoh, Secaas implementation guidance: Network security implementation guidance, Cloud Security Alliance, 2013, available at: https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_10_Network_Security_Implementation_Guidance.pdf, 2013.

31 V.I. Munteanu, C. Mindruta and T. Fortis, Service Brokering in Cloud Governance, 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), pp.497-504, 2012.

32 T. Han and K. M. Sim,, An Ontology-enhanced Cloud Service Discovery System, International Multi-Conf. Engineers and Computer Scientists (IMECS), vol. 1, 2010.

33 Cloud Computing Use Cases group, Cloud Computing Use Cases White Paper, available at: http://opencloudman ifesto.org/Cloud_Computing_Use_Cases_Whitepaper-4_0.pdf, 2014.

34 M.A. Rodríguez-García, R. Valencia-García, F. García-Sánchez and J.J. Samper-Zapater, Ontology-based annotation and retrieval of services in the cloud, Knowledge-Based Systems, vol. 56, pp. 15-25, 2014.

35 M. Serrano, L. Shi, M.Ó. Foghlú and W.Donnelly, Cloud Services Composition Support by Using Semantic Annotation and Linked Data, Knowledge Engineering and Knowledge Management, Communications in Computer and Information Science, vol. 348, pp 278-293, 2013.

36 V. Presutti and A. Gangemi, A Library of Ontology Design Patterns: reusable solutions for collaborative design of networked ontologies. NeOn project deliverables D2.5.1, available at: http://www.neon-project.org/deliverables/WP2/NeOn_2008_D2.5.1.pdf, 2008.

37 A. Gangemi and V. Presutti, Ontology Design Patterns, Handbook on Ontologies, International Handbooks on Information Systems, Springer-Verlag Berlin Heidelberg, pp. 221-243, 2009.

38 D. Vrande˘ci´c, Y. Sure, R. Palma and F. Santana, Ontology Repositories and Content Evaluation, Knowledge Web project deliverables D1.2.10v2, available at: http://knowledgeweb.semanticweb.org/semanticportal/deliverables/D1.2.10v2.pdf, 2007.

39 M. Horridge, A practical guide to building OWL ontology's using protégé 4 and CO-ODE tools, The University of Manchester, edition 1.3, available at http://owl.cs.manchester.ac.uk/tutorials/protege owltutorial/resources/ProtegeOWLTutorialP4_v1_3.pdf, 2011.

40 Dropbox, Inc., Dropbox Platform developer guide, available at: https://www.dropbox.com/developers/reference/devguide, 2014.

41 E. Portmann, A. Meier, P. Cudré-Mauroux and W. Pedrycz, FORA — A Fuzzy Set Based Framework for Online Reputation Management, Fuzzy Sets and Systems, 2014.

42 Amazon Web Services, Inc., Amazon S3 Developer Guide, available at: http://awsdocs.s3.amazonaws.com/S3/latest/s3-dg.pdf, 2014.

43 D. Tsarkov and I. Horrocks, FaCT++ description logic reasoner: system description, 3ird international joint conference on Automatic Reasoning, pp. 292-297, 2006.

44 R. Shearer, B. Motik and I. Horrocks, HermiT: a highly efficient OWL reasoner, 5th OWL Experienced and Directions Workshop, pp. 26-27, 2008.

45 M. Horridge and S. Bechhofer, The OWL API: A Java API for OWL Ontologies Semantic Web Journal 2(1), Special Issue on Semantic Web Tools and Systems, pp. 11-21, 2011.

46 Google, Inc., Overview of App Engine Features, available at: https://developers.google.com/appengine/features/, 2014.

47 Policy-Based Automated Scaling for IBM SmarCloud Application Services, available at: http://www.ibm.com/cloud-computing/uk/en/paas.html, 2013

48 M. Sabou and M. Fernandez, Ontology (network) evaluation. Ontology Engineering in a Networked World, Springer, pp. 193-212, 2012.

49 M. Rico, M.L. Caliusco, O. Chiotti and M.R. Galli, OntoQualitas: A framework for ontology quality assessment in information interchanges between heterogeneous systems, Computers in Industry, vol. 65, no. 9, pp. 1291-1300, 2014.

50 A. Burton-Jones, V.C. Storey, V. Sugumaran and P. Ahluwalia, A semiotic metrics suite for assessing the quality of ontologies, Data & Knowledge Engineering, vol. 55, no. 1, pp. 84-102, 2005.

51 D. Fang, X. Liu, I. Romdhani and C. Pahl, An approach to unified cloud service access, manipulation and dynamic orchestration via semantic cloud service operation specification framework, Journal of Cloud Computing, vol. 4, no. 14, available at: http://www.journalofcloudcomputing.com/content/4/1/14, 2015.

52 C. Pahl, F. Fowley, P. Jamshidi, D. Fang and X. Liu, A classification and comparison framework for cloud service brokerage architectures, IEEE Transactions on Cloud Computing, under review, 2015.

53 H.N. Talantikite, D. Aissani and N. Boudjlida, Semantic annotations for web services discovery and composition, Computer Standards & Interfaces, vol. 31, no. 6, pp. 1108-1117, 2009.

54 J. Sangers, F. Frasincar, F. Hogenboom and V. Chepegin, Semantic Web service discovery using natural language processing techniques, Expert Systems with Applications, vol. 40, no. 11, pp. 4660-4671, 2013.

55 H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, C. Ursu, M. Dimitrov, M. Dowman, N. Aswani, I. Roberts, Y. Li, Developing Language Processing Components with GATE Version 5: (a User Guide). University of Sheffield, available at: http://gate.ac.uk /releases/gate-5.0-build3244ALL/doc/tao/splitch1.html, 2013.

56 G.D. Modica, G. Petralia and O. Tomarchio, A Business Ontology to Enable Semantic Matchmaking in Open Cloud Markets, 8th International Conference on Semantics, Knowledge and Grids (SKG), pp. 96-103, 2012.

57 K. Tserpes, F. Aisopos, D. Kyriazis and T. Varvarigou, A Recommender Mechanism for Service Selection in Service-oriented Environment, Future Generation Computer Systems, vol. 28, no. 8, pp. 1285-1294, 2012.

58 F. Moscato, F. Fortis, V. Munteanu, R. Aversa and D. Petcu, Cloud ontology and Cloud resources representations, mOSAIC project deliverables D1.2, 2011.

59 F. Bobillo, U. Straccia, An OWL ontology for fuzzy OWL2, Foundations of Intelligent Systems, Lecturer notes in Computer Science, vol. 5722, pp. 151-160, 2009.

60 G. Stoilos, G. Stamou and J.Z. Pan, Fuzzy extensions of OWL: Logical properties and reduction to fuzzy description logics, International J. Approximate Reasoning, vol. 51, no. 6, pp. 656-679, 2010.

Daren Fang is a PhD student and research assistant at Edinburgh Napier University. His research interests include cloud service modeling, knowledge engineering, green service optimization, service adaptation, and service evolution.

Dr. Xiaodong Liu is a reader and the director of Centre for Information & Software Systems in School of Computing at Edinburgh Napier University. His research interests include context-aware adaptive services, service evolution, mobile clouds, pervasive computing, software reuse, and green software engineering. He is a member of IEEE Computer Society and British Computer Society.

Dr. Imed Romdhani is a Lecturer in Networking at Edinburgh Napier University. He obtained a M.Sc. in networking from Louis Pasteur University of Strasbourg (ULP), France in 2001. He received his PhD degree from the University of Technology of Compiegne, France in 2005. He was a research engineer with Motorola Labs Paris from 2001 to 2005. His research interests include cloud-based networking, mobile IP, moving network, mesh networks, and IP security.

Dr. Claus Pahl is a senior lecturer in Dublin City University. He published over 200 papers on a range of software engineering, including cloud and services system engineering. He has been involved in a number of national and international research projects, and as a PI, received research grants with a total awarded funding of € 2.58 million.

Pooyan is a postdoctoral research fellow at Dublin City University, currently working on cloud migration and trust and dependability aspects of cloud systems within IC4 – the Irish Centre for Cloud Computing and Commerce. His key research interest is the dynamic evaluation of quality metrics in software systems and environments such as the cloud

Daren Fang is a PhD student and research assistant at Edinburgh Napier University. His research interests include cloud service modeling, knowledge engineering, green service optimization, service adaptation, and service evolution.

Dr. Xiaodong Liu is a reader and the director of Centre for Information & Software Systems in School of Computing at Edinburgh Napier University. His research interests include context-aware adaptive services, service evolution, mobile clouds, pervasive computing, software reuse, and green software engineering. He is a member of IEEE Computer Society and British Computer Society.

Dr. Imed Romdhani is a Lecturer in Networking at Edinburgh Napier University. He obtained a M.Sc. in networking from Louis Pasteur University of Strasbourg (ULP), France in 2001. He received his PhD degree from the University of Technology of Compiegne, France in 2005. He was a research engineer with Motorola Labs Paris from 2001 to 2005. His research interests include cloud-based networking, mobile IP, moving network, mesh networks, and IP security.

Dr. Claus Pahl is a senior lecturer in Dublin City University. He published over 200 papers on a range of software engineering, including cloud and services system engineering. He has been involved in a number of national and international research projects, and as a PI, received research grants with a total awarded funding of € 2.58 million.

Pooyan is a postdoctoral research fellow at Dublin City University, currently working on cloud migration and trust and dependability aspects of cloud systems within IC4 – the Irish Centre for Cloud Computing and Commerce. His key research interest is the dynamic evaluation of quality metrics in software systems and environments such as the cloud.

Daren Fang
Click here to download high resolution image

Xiaodong Liu
Click here to download high resolution image

Imed Romdhani
Click here to download high resolution image

Claus Pahl
Click here to download high resolution image

Pooyan Jamshidi
Click here to download high resolution image